Tutorial 3 Solutions

1

(a) **The method computeSum() computes the sums of all the elements in the subarrays and the array that begin at index 0. In this example, it computes the following:**

Sum = 5 for the subarray [5]
Sum = 9 for the subarray [5, 4]
Sum = 12 for the subarray [5, 4, 3]
Sum = 23 for the subarray [5, 4, 3, 11]
Sum = 32 for the array [5, 4, 3, 11, 9]

(b) **In the inner for loop, the statement sum += item[j] is executed i times where i $\in$ {1, 2, …, n}. In this example, n = e.length = 5.**

At the 1st inner for loop, i = 0, sum += item[j] is not executed.
At the 2nd inner for loop, i = 1, sum += item[j] is executed once.
.
.
At the (n-1)-th inner for loop, i = n-2, sum += item[j] is executed n-2 times.
At the n-th inner for loop, i = n-1, sum += item[j] is executed n-1 times.

Therefore, sum += item[j] is executed $\sum_{i=1}^{n-1} i = n(n-1)/2$ = $n^2/2 - n/2$ **times.**
Thus, number of step count is $n^2/2 - n/2$.
The big-O notation is $O(n^2)$.

There are 3 operations for each sum += item[j] is executed,
Thus, number of operation is $3n^2/2 - 3n/2$.
The big-O notation is $O(n^2)$.

(c) **Sum for array 0 is 5**
Sum for array 1 is 9
Sum for array 2 is 12
Sum for array 3 is 23
Sum for array 4 is 32

2

(a) `Answer: O(n²)`
(b) `Answer: O(log n)`
(c) `Answer: O(n Log n)`

Explanation: If you notice, j keeps doubling till it is less than or equal to n. Number of times, we can double a number till it is less than n would be log(n).
Let's take the examples here.
for n = 16, j = 2, 4, 8, 16
for n = 32, j = 2, 4, 8, 16, 32
So, j would run for O(log n) steps.
i runs for n/2 steps.
So, total steps = O(n/ 2 * log (n)) = **O(n*logn)**

3.

(a) **Suppose** $p(a,b) = \sqrt[3]{18}a^3b + a^4b\lg(a) + \frac{\sqrt{a}}{11}ab\lg(a) + \lg(109)$

The big-O notation is $O(a^4b\lg(a))$.

(b) **Suppose** $f(n) = 31n^{2.5} + \sqrt{19} + 7n^2\log(n)$

The big-O notation is $O(n^{2.5})$.